Modelo Temporizado de Exclusión para Grupos de Procesos

Karina M. Cenci * María Cecilia Reyes **

Laboratorio de Investigación en Sistemas Distribuidos
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Resumen

In distributed systems, the applications can compete in the use of a resource or can work together and need a resource. To keep these requirements its necessary a protocol to guarantee the access to the shared resources. Depending on the kind of problem, we can solve through the traditional algorithms of mutual exclusion or through the extensions of k-process or group of process. In this article, we consider the mutual exclusion problem for group of process, providing the conditions of mutual exclusion and concurrency, minimizing the answer time in centralized and distributed systems with time restrictions.

Keywords: Mutual Exclusion - Group Mutual Exclusion - Concurrency - Time Restriction - Distributed Systems

Resumen

En sistemas distribuidos las aplicaciones realizan trabajos que requieren acceso en forma exclusiva a un recurso o realizan trabajo en forma conjunta para el cual requieren de la utilización de un recurso, para poder mantener estos requerimientos es necesario contar con protocolos que garanticen el acceso a los recursos compartidos. Dependiendo del problema se puede resolver a través del tradicional algoritmo de exclusión mutua, o a través de las extensiones a k o grupo de procesos. En este trabajo, se considera el problema de exclusión mutua para grupos de procesos, que garantice las condiciones de exclusión mutua y concurrencia, minimizando el tiempo de respuesta en ambientes centralizados y distribuidos con restricciones de tiempo.

Palabras Claves: Exclusión Mutua - Exclusión Mutua de Grupo - Concurrencia - Restricciones de Tiempo - Sitemas Distribuidos

* e-mail: kmc@cs.uns.edu.ar
** e-mail: creyes@criba.edu.ar

1.- Introducción

En algunas aplicaciones es necesario garantizar concurrencia y a su vez exclusión mutua en el uso de los recursos. En algunos casos se resuelve el problema a través de la utilización de un protocolo que garantice exclusión mutua a un único proceso del sistema, este problema corresponde al problema tradicional, en el cuál un sólo proceso accede a la sección crítica. En el caso de los sistemas distribuidos, hay algunas aplicaciones tales como las que soportan trabajo cooperativo, es necesario imponer exclusión mutua sobre un conjunto de procesos, mientras que los procesos del mismo grupo comparten.

En la literatura existen una gran variedad de soluciones para este problema [4], [9], [10], [11], [12]. La idea de este trabajo es alcanzar un protocolo, que garantice las condiciones de exclusión mutua y concurrencia, minimizando el tiempo de respuesta en ambientes centralizados y distribuídos con restricciones de tiempo. El resto del trabajo está organizado como sigue. La sección 2 presenta una breve descripción de los tipos de algoritmos disponibles para el tratamiento de exclusión mutua. La sección 3 expone el modelo base del problema que se pretende resolver. En la sección 4 se describe el modelo de la solución propuesta y en la sección 5 se presentan los algoritmos para ambientes mono y multiprocesador. En este trabajo se utiliza en forma indistinta nodo y procesador.

2.- Preliminares

Los algoritmos de exclusión mutua que utilizan el modelo de memoria compartida, se los puede clasificar de acuerdo a las propiedades que presentan como: algoritmos de exclusión mutua *rápidos*, algoritmos *adaptivos*, algoritmos de exclusión mutua basados en el tiempo, algoritmos *no atómicos*.

En los algoritmos de exclusión mutua *rápidos*[10], existe una gran diferencia en la complejidad de tiempo entre casos que están libres de contención y en los que se presenta contención. Esto significa que si la sección crítica está disponible, el recurso no está siendo utilizado, la complejidad de tiempo será constante, y en los otros casos la complejidad podrá o no tener puntos de contención.

En los algoritmos de exclusión mutua *adaptivo*, el incremento en la complejidad de tiempo cuando hay contención es más gradual, está en función del número de procesos en contención. En un algoritmo adaptable la complejidad del paso de una operación depende solamente del número de procesos que realmente ejecutan el paso concurrentemente con esa operación, esto es, la complejidad de una operación está en función de la contención real que encuentra y no del número total de procesos. Las propiedades consideradas para la implementación de un algoritmo adaptable están relacionadas con el nivel de contención y de adaptabilidad. Estas características también se las puede aplicar a las extensiones del problema de exclusión mutua, para grupos de procesos, para k procesos.

Otro factor importante para determinar la velocidad de un algoritmo es la cantidad de tráfico de interconexión que el genera. En función de este otro parámetro se define a la *complejidad de tiempo* de un algoritmo de exclusión mutua a ser el peor caso en el número de referencias de memoria remotas por un proceso en orden para ingresar y salir de su sección crítica. En los modelos asincrónicos, no se considera un límite en el tiempo requerido para que una instrucción se ejecute. En los algoritmos basado en tiempo se considera un tiempo límite de ejecución para las instrucciones, pueden presentarse 2 modelos: demora-conocida ó demora-desconocida. Existen otros algoritmos en los cuales el tiempo está relacionado con el proceso que quiere acceder a la sección crítica, en este caso el tiempo se lo considera como una prioridad y se lo

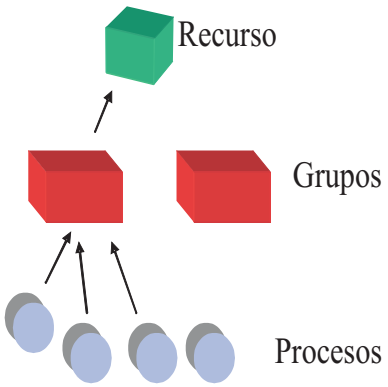

Figura 1: *Exclusión Mutua para grupo de procesos*

utiliza en el protocolo de selección en la sección de entrada.

3.- Modelo Base del Problema

El problema presenta dos tipos de actores: los *procesos* y los *grupos*, que se integran para competir por la utilización de un recurso. El modelo presentado en [8], tiene dos componentes que forman parte del mismo, el proceso que selecciona un grupo de trabajo, y el grupo que compite para acceder a la sección crítica. En las soluciones al problema de exclusión mutua, la ejecución de los procesos respeta un ciclo que está compuesto por las secciones: *crítica, de entrada, de salida* y *resto*.

En la figura 1 se muestra la relación entre los actores del problema.

Cuando un actor no está involucrado de ninguna manera con el recurso, se dice que está en la sección *resto*. Para obtener la admisión a la sección crítica, el actor ejecuta un protocolo de entrada (*trying*), después que utiliza el recurso, se ejecuta un protocolo de salida (*exit*). Este procedimiento puede repetirse, de modo que cada actor sigue un ciclo, desplazándose desde la *sección resto (R)*, a la *sección de entrada (T)*, luego a la *sección crítica (C)* y por último a la *sección de salida (E)*, y luego vuelve a comenzar el ciclo en la *sección resto*.

Como se observa en la figura 2, el primer paso que realiza el *actor proceso*, en la sección de entrada, es seleccionar el grupo en el cual desea participar del conjunto de m grupos. El segundo paso es esperar hasta que el grupo seleccionado entre en la sección crítica para que pueda acceder a la misma. Cuando finaliza su actividad, sale de la sección crítica, se desvincula del grupo (sección de salida).

El *actor grupo* inicialmente está inactivo, en la sección resto, esto representa que ningún proceso lo ha seleccionado para participar en el mismo. El primer proceso que lo selecciona para participar, hace que comience la competencia por entrar en la sección crítica, y se lo identifica como el primer proceso que pertenece al grupo; pasa a la sección de entrada. Todos los procesos que lo seleccionen mientras se encuentra en competición por entrar a la sección crítica, se agregan a los procesos ya existentes. En el caso que el grupo esté en la sección crítica, si el proceso que activó al grupo está trabajando en la misma, entonces el proceso se incorpora, sino se pone en cola de espera hasta que termine la actual vuelta (todos los procesos que están trabajando finalicen su tarea y el grupo salga de la sección crítica), se reinicie el ciclo, esto es, compita nuevamente por el ingreso en la sección crítica. En la figura 3 se observa el comportamiento del actor grupo mientras se encuentra activo.

Los *actores grupos* compiten por alcanzar el permiso para acceder al recurso (acceso a la

Proceso$_i$

1. ..... {Sección Resto}
2. El proceso selecciona el grupo de trabajo {Grupo$_k$}
3. Espera hasta que entra a la sección crítica {Sección de Entrada}
4. ..... { Sección Crítica}
5. Sale de la sección crítica y se desvincula del grupo.
6. ..... {Sección Resto}

n

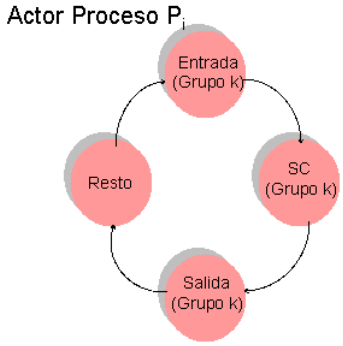

Figura 2: *Esquema 1*

Grupo$_k$
1.- ..... {Sección Resto}
2.- Un proceso lo selecciona para trabajar en él.
{Sección de Entrada}
3.- Si es el primer proceso en el grupo entonces
Comienza a competir en el ingreso a la S.C.
4.- sino
Si no está en la S.C. entonces
Agrega el proceso a la lista de procesos
sino
Si está el primero del Grupo entonces
entra en la S.C. el proceso
sino
El proceso lo coloca en la lista de espera
hasta que termine la sección crítica
actual y compita por ingresar nuevamente
5.- ..... {Sección Resto}

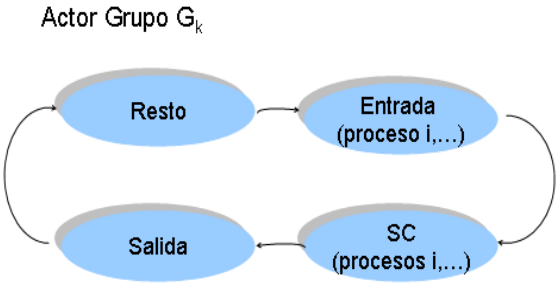

Figura 3: *Esquema 2*

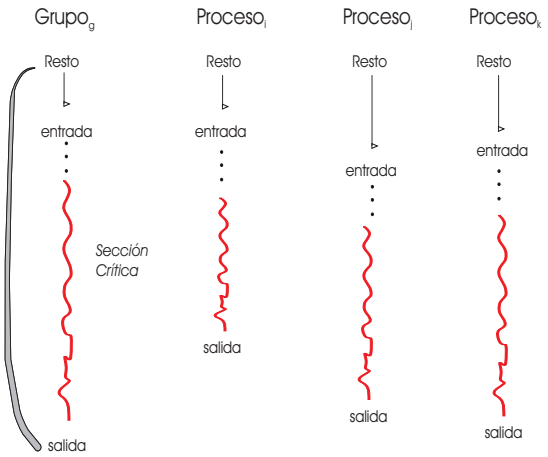

Figura 4: Concurrencia en el grupo g

región crítica), y sólo un único grupo tiene derecho de utilizar el recurso en un determinado instante de tiempo.

La idea es que el algoritmo cumpla las siguientes condiciones:

- Un único grupo está activo utilizando el recurso compartido (exclusión mutua)
- Si el recurso está disponible y un grupo quiere utilizarlo (está en espera) que acceda al mismo sin tener más demora.
- Si un grupo está activo y el primer proceso también, todo proceso que quiera trabajar en el mismo que lo pueda realizar, de esa manera se logra un mayor nivel de concurrencia.

Como se observa en la figura 4, el grupo g se encuentra en la región crítica, el proceso i es el primer proceso del grupo, luego le sigue el proceso k que se agrega al grupo, entran a la región crítica en forma concurrente, cuando el proceso j selecciona el grupo g, como este ya se encuentra en la región crítica y el primer proceso (proceso i) sigue trabajando en la misma entonces puede acceder también, sin tener que esperar y trabajar concurrentemente.

4.- Modelo del Temporizado

EDF (Earliest Deadline First) [2] [3] es uno de los modelos utilizados en planificaciones de tiempo real duro, que también puede emplearse en aplicaciones de tiempo real blando. Este algoritmo de planificación se adapta a tareas periódicas, aperiódicas y esporádicas. La versión original del protocolo es apropiativa, pero también puede utilizarse en forma no apropiativa. A diferencia de RMA (Rate Monotonic Algorithm) [2], la asignación de prioridades es dinámica, basada en la urgencia de una tarea, por lo tanto la tarea más urgente obtiene la máxima prioridad. Este modelo requiere:

- deadlines absolutos, esto significa, el deadline relativo original incrementado por el tiempo de arribo de la tarea.
- una cola de tareas ordenada en forma creciente por deadline absoluto.

El factor de utilización de una tarea T_i se define como E_i/P_i, donde E_i es el tiempo de ejecución de la tarea y P_i su período.

El factor de utilización del conjunto de n tareas T se define como

$$U = \sum_{i=1}^{n} (E_i/P_i). \quad (1)$$

El test de viabilidad para n tareas periódicas es el siguiente:

$$\sum_{i=1}^{n} (E_i/P_i) <= 1. \quad (2)$$

Cuando la planificación incluye tanto tareas aperiódicas como periódicas el objetivo es reducir el tiempo de respuesta de las aperiódicas tanto como sea posible garantizando aún que todas las periódicas cumplan con sus deadlines. Si U es la utilización del procesador con tareas periódicas, la fracción completa restante $1-U$ puede ser siempre alocada a la ejecución de tareas aperiódicas cuando se utiliza una planificación EDF, es por esto que esta política obtiene una performance superior a otros métodos de planificación de tareas aperiódicas.

5.- Algoritmo Temporizado EM para Grupos de Procesos

Las soluciones propuestas en la literatura al problema presentado en 3.-, varían en el entorno en que han sido consideradas, centralizado o distribuido, y en los objetivos a optimizar planteados, principalmente en lo que se refiere a maximizar el grado de concurrencia o minimizar la espera de aquellos procesos que solicitan acceso a un grupo diferente al activo. En alguna medida, estos objetivos se contraponen, como se verá más adelante, y se hace necesario optar por alguno de ellos cuando se plantea una solución. En esta sección se propone inicialmente una solución para entornos monoprocesador, donde los procesos tienen restricciones temporales blandas, y como consecuencia se pretende asegurar una espera limitada para aquellos procesos que compiten para acceder al recurso. Posteriormente, esta solución es extendida para poder ser empleada en entornos multiprocesador con memoria compartida.

En la figura 5 se presenta el algoritmo propuesto para entornos centralizados. Cuando un proceso quiere vincularse a un grupo para utilizar un recurso en forma concurrente, selecciona el grupo y verifica si puede ser incluido para competir por al acceso al recurso respetando las restricciones temporales de los procesos del resto de los grupos. Es decir se controla si la aceptación de este nuevo proceso permite cumplir los deadlines de los demás (test de viabilidad). Si no fuera así, el proceso no es aceptado en esta vuelta, y deberá reintentar en otro pasaje. Si es posible cumplimentar todos los deadlines, el proceso activa el grupo si éste está inactivo, y espera en caso en que el recurso esté asignado a un grupo diferente, o ingresa si el recurso estaba disponible. Si el grupo está activo, ya sea compitiendo por el recurso o haciendo uso de él, el proceso que desea unirse debe verificar que su deadline absoluto sea similar al del proceso que activó el grupo y que es considerado como referente de la sesión actual del mismo. Esto se debe a que se permite integrar un grupo a aquellos procesos que lo requieran y que se prevea su finalización en un tiempo próximo a la finalización prevista para el referente. De manera que se pueda asegurar una espera limitada para aquellos procesos que compiten por el recurso en otros grupos. Si el deadline del proceso no se encuentra próximo al del referente, el proceso no es aceptado en esta vuelta, y deberá reintentar en otro pasaje. La proximidad de los deadlines de los procesos de un grupo es un parámetro del mismo y es inversamente proporcional al grado de concurrencia que se permite en el grupo. Cuanto más próximos se encuentren los

Para un proceso P_k, (r_k, e_k, D_k) con $d_k = r_k + D_k$, cada vez que éste requiere ingreso al grupo G_i debe verificar:

```
Repetir hasta ingresar o no satisfacer deadline del proceso
    Si (d1i <= dk <= d1i * (1 + GRi)) o (d1i = 0)

    /* si Pk es el primer proceso en requerir unirse al grupo Gi entonces d1i = 0 y si es aceptado para ingresar se
    activará el grupo con d1i = dk, que obviamente verifica la condición anterior. */

    Entonces
        Para todos los grupos Gj ; 1<= j <= m
            Dj = D1j * (1 + GRj); /* deadline grupal */
            Para todos los procesos Pl en la cola del grupo Gj, calcular:
                Ej= Ej + el;
            Si (i = j)
            Entonces
                Ej= Ej + ek;
            U = U + Ej / Dj;
        Si U <= 1
            Entonces
                Poner en cola del grupo Gi a Pk
                Si el grupo Gi está activo
                Entonces
                    Ingresar el proceso al grupo.
                Sino
                    Activar el grupo con d1i=dk
                    Ingresar el proceso al grupo
```

Al finalizar el proceso la ejecución de su sección crítica y egresar del grupo G_i, es retirado de la cola del mismo y si éste era el último, el grupo es desactivado y el d_{1i}=0.

Figura 5: Esquema del Algoritmo

deadlines, menos procesos accederán en forma concurrente, pero el tiempo de espera del resto de los procesos de otros grupos será menor. Esta situación es a la que se hacía referencia cuando se mencionaba la contraposición de objetivos planteados en las distintas soluciones propuestas.

Más formalmente, el problema anterior puede ser modelado utilizando una planificación a nivel de grupo EDF no apropiativa, donde la exclusión mutua entre los grupos de procesos está impuesta por la misma política de planificación, se recuerda que el recurso es único y no apropiable.

Para este modelado, se ha seleccionado y adaptado un algoritmo de planificación de tiempo real no apropiativo propuesto en [1] conocido como gEDF o group-EDF. Este algoritmo se basa en el agrupamiento dinámico de procesos con deadlines muy próximos, efectuando una planificación EDF entre los grupos y SJF (Shortest Job First) dentro de cada uno de ellos. Para los fines de este trabajo, la política de planificación interna de cada grupo es irrelevante por lo que ha sido dejada libre, para permitir la selección posterior en función de otros aspectos de interés a las aplicaciones en particular. De acuerdo a los objetivos fijados al comienzo de esta sección, el algoritmo seleccionado se adapta particularmente bien a la naturaleza del problema y permite simplificar el tratamiento del mismo. Optimiza los tiempos de respuesta de los procesos limitando la cantidad de ellos que pueden adherir a un grupo, y con la utilización de una planificación no apropiativa que reduce el overhead necesariamente introducido por los cambios de contexto.

5.1. - Algoritmo Temporizado Centralizado

Este modelo está caracterizado por:

1. Un conjunto de procesos de tiempo real blando P_k (r_k, e_k, D_k), donde r_k, e_k y D_k representan su tiempo de arribo, ejecución y deadline relativo respectivamente.

2. Un conjunto de grupos G ={G_1,G_2, ..., G_m}., donde cada grupo G_i (GR_i) tiene asignado un parámetro de rango de grupo GR_i >=0, que representa la tolerancia del deadline para todos los procesos del grupo.

3. Cuando un proceso P_k solicita unirse a un grupo G_i, debe verificar dos condiciones:

   - $d_1 <= d_k <= d_1 * (1 + GR_i)$; donde d_1, d_k representan los deadlines absolutos de los procesos P_1 y P_k respectivamente. P1 corresponde al proceso que activó el grupo, o sea el primer proceso que solicitó el acceso, y es considerado como elemento referente de la sesión actual del grupo. Esta condición establece que podrán ingresar en forma concurrente al grupo todos aquellos procesos cuya ejecución deba finalizar como máximo cuando finalice el plazo del elemento referente, y contribuye a garantizar espera limitada ya que el elemento referente puede permanecer en el grupo un tiempo finito.
   - Test de viabilidad:

$$\sum_{i=1}^{m}(E_i/D_i) <= 1;\ donde\ D_i = D_{1i} * (1 + GR_i) \tag{3}$$

     El test de viabilidad considera para cada grupo el total de tiempo de ejecución acumulado por los procesos del mismo, y su relación con el deadline grupal, de manera de garantizar que todos los grupos cumplan sus restricciones de tiempo.

4. Si el proceso no verifica las condiciones de 3. entonces no es aceptado en la vuelta presente y deberá reintentar en otro pasaje.

5. Si verifica las condiciones y el grupo no está activo, lo activa. Luego, si el recurso está disponible o el grupo ya estaba utilizándolo, comienza a ejecutar inmediatamente. Aquí interviene la política de planificación interna asociada al grupo. Si el recurso no estuviera disponible o el grupo aún estuviera compitiendo por su acceso, espera en una cola asociada al grupo, hasta que el algoritmo g-EDF lo seleccione de acuerdo a su $d_i = d_{1i} * (1 + GR_i)$.

En la figura 6 se muestra el comportamiento de cada proceso, el proceso P_1 es el primer proceso que se asocia con el grupo G_i, donde $d_1 * (1 + GR_i)$ es el tiempo máximo que el grupo G_i utilizará el recurso durante esta sección crítica y d_1 es el tiempo máximo que P_1 estará asociado al grupo. Como $d_1 < d_1 * (1 + GR_i)$; cuando el grupo accede al recurso este comienza su utilización. En el momento que el proceso P_k se asocia al grupo con un deadline absoluto d_k, éste se encuentra utilizando el recurso (está en la sección crítica), como el tiempo máximo que estará vinculado es menor que el tiempo que le resta al grupo permanecer en la sección crítica, esto es $d_k < d_1 * (1 + GR_i)$, entonces puede utilizar en forma concurrente el recurso. Cuando el proceso P_l se asocia al grupo con un deadline absoluto d_l, también controlará el tiempo que requiere utilizar el recurso con el tiempo que el grupo le resta utilizar el mismo, en este caso se tiene que $d_l \not< d_1 * (1 + GR_i)$, por lo tanto el proceso P_l es demorado hasta la próxima vuelta, cuando el grupo G_i vuelva a acceder al recurso.

5.2. - Algoritmo Temporizado Distribuido

El modelo presentado en 5.1.- ha sido extendido para poder ser empleado en entornos distribuidos considerando:

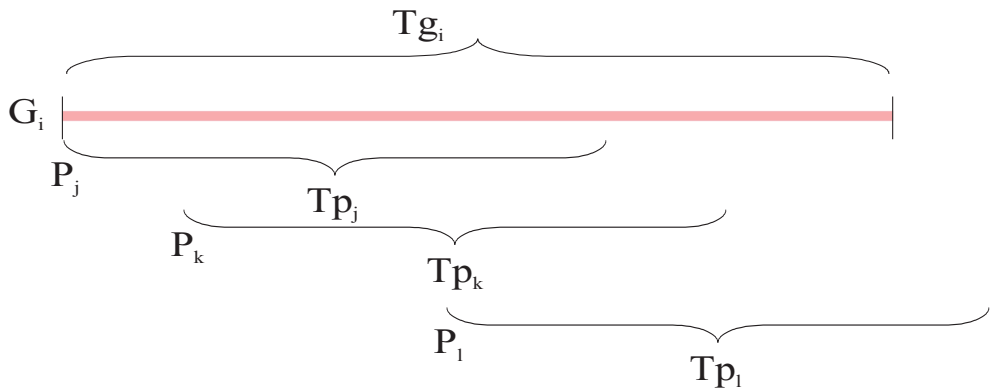

Figura 6: Asociación de Grupos y Procesos

1. Un conjunto de procesadores H=$\{H_1, H_2, ..., H_p\}$ que comparten memoria.

2. Un conjunto de procesos de tiempo real blando P_k (r_k, e_k, D_k, H_k), donde r_k, e_k, D_k, y H_k representan su tiempo de arribo, ejecución, deadline relativo, y procesador en el que ejecuta respectivamente.

3. Un conjunto de grupos G =$\{G_1, G_2, ..., G_m\}$., donde cada grupo G_i (GR_i) tiene asignado un parámetro de rango de grupo GR_i >=0, que representa la tolerancia del deadline para todos los procesos del grupo.

4. Cuando un proceso P_k solicita unirse a un grupo G_i, debe verificar dos condiciones:

   - $d_1 <= d_k <= d_1 * (1 + GR_i)$; donde d_1, d_k representan los deadlines absolutos de los procesos P_1 y P_k respectivamente. P1 corresponde al proceso que activó el grupo, o sea el primer proceso que solicitó el acceso, y es considerado como elemento referente de la sesión actual del grupo. Esta condición establece que podrán ingresar en forma concurrente al grupo todos aquellos procesos cuya ejecución deba finalizar como máximo cuando finalice el plazo del elemento referente, y contribuye a garantizar espera limitada ya que el elemento referente puede permanecer en el grupo un tiempo finito.
   - Test de viabilidad:

$$\sum_{i=1}^{m} max(E_{iH1}, E_{iH2}, ..., E_{iHp})/D_i <= 1; \quad (4)$$

     donde $D_i = D_{1i} * (1 + GR_i)$ y E_{iHj} representa el tiempo de ejecución total acumulado por los procesos del grupo G_i que ejecutan en el procesador H_j.

5. Si el proceso no verifica las condiciones de 4. entonces no es aceptado en la vuelta presente y deberá reintentar en otro pasaje.

6. Si verifica las condiciones y el grupo no está activo, lo activa. Luego, si el recurso está disponible o el grupo ya estaba utilizándolo, comienza a ejecutar inmediatamente. Si el recurso no estuviera disponible o el grupo aún estuviera compitiendo por su acceso, espera en la cola del grupo asociada al procesador que solicita, hasta que el algoritmo g-EDF lo seleccione de acuerdo a su $d_i = d_{1i} * (1 + GR_i)$.

Las variables E_{iHj} y D_i , 1<=i<=m , 1<=j<=p, y estructuras de datos que implementan las colas asociadas a cada procesador y grupo, son compartidas en memoria y deben ser accedidas atómicamente en forma exclusiva. Para esto es posible utilizar cualquiera de las herramientas disponibles para asegurar exclusión mutua de procesos en su forma tradicional.

5.3.- Casos de Aplicación

5.3.1.- Caso 1

Se tiene un Congreso que está formado por diferentes Foros y existe una única sala de conferencias para que se realicen los foros. En cada uno de los foros disertan filósofos que se asocian a los mismos. Un foro tiene asociado un tiempo máximo que puede permanecer en la sala de conferencias, para que lo puedan utilizar el resto de los foros. Cada filósofo tiene asociado un tiempo que requiere para su disertación en el foro, para poder ingresar el tiempo que requiere debe verificar que sea menor que el tiempo que le queda en la sala de conferencias. Se tiene la siguiente situación, hay 4 foros F_1, F_2, F_3 *y* F_4, hay 10 filósofos $f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9$ *y* f_{10}.

Para este caso se utiliza el algoritmo centralizado, que se presenta en la sección 5.1. Inicialmente están todos los foros inactivos. Consideramos el rango del grupo estático, para todo i, GR_i=3, como simplificación para este ejemplo.

- El filósofo f_1 se asocia al foro F_1, con (0, 2, 5) representando cada uno de los parámetros respectivamente (r_1, e_1, D_1). El $d_1 = r_1 + D_1$, $d_1 = 5$.
- Cuando se asocia f_1 al foro F_1, activa el foro y se asigna a $DF_1 = d_1 * (1 + GR_1) = 8$. Como es el único foro que quiere utilizar la sala de conferencias puede ingresar en la misma.
- El filósofo f_2 se asocia al foro F_1, con (1, 2, 5) representando a (r_2, e_2, D_2). El $d_2 = r_2 + D_2$, $d_2 = 6$.
- Cuando se asocia f_2 al foro F_1, el foro ya se encuentra activo. Como cumple con la condición $d_1 < d_2 < DF_1$, entonces realiza el test de viabilidad y le da como resultado que $U = 1/2$ por lo tanto es aceptado en el foro.
- El filósofo f_3 se asocia al foro F_2, con (2, 4, 5)representando a (r_3, e_3, D_3). El $d_3 = r_3 + D_3$, $d_3 = 7$.
- Cuando se asocia f_3 al foro F_2, activa el foro y se asigna a $DF_2 = d_3 * (1 + GR_2) = 10$. Se realiza el test de viabilidad con respecto a los otros grupos para ver si se lo acepta o rechaza. Al realizar el test se tiene que $E_1 = 4$, $E_2 = 4$, $DF_1 = 8$, $DF_2 = 10$, el valor de $U = 9/10$ por lo tanto es aceptado.
- El filósofo f_4 se asocia al foro F_2, con (3, 2, 4) representando a (r_4, e_4, D_4). El $d_4 = r_4 + D_4$, $d_4 = 7$.
- Cuando se asocia f_4 al foro F_2, el foro ya se encuentra activo. Como cumple con la condición $d_3 < d_4 < DF_2$, entonces realiza el test de viabilidad y le da como resultado que $U = 11/10$ por lo tanto no es aceptado en el foro.

5.3.2.- Caso 2

Se tiene un Congreso que está formado por diferentes Foros y existe una única sala de conferencias para que éstos se realicen. En cada foro participan filósofos que debaten sobre el tema que propone en el mismo. Todos los filósofos cuando participan del foro están activos desde el momento que ingresan.

Para este caso se utiliza el algoritmo distribuido, presentado en la sección 5.2. Inicialmente están todos los foros inactivos. Consideramos el rango del grupo estático, para todo i, GR_i=3, como simplificación para este ejemplo.

- El filósofo f_1 se asocia al foro F_1, con (0, 2, 5, 1) representando cada uno de los parámetros respectivamente (r_1, e_1, D_1, H_1). El $d_1 = r_1 + D_1$, $d_1 = 5$.
- Cuando se asocia f_1 al foro F_1, activa el foro y se asigna a $DF_1 = d_1 * (1 + GR_1) = 8$. Como es el único foro que quiere utilizar la sala de conferencias puede ingresar en la misma.
- El filósofo f_2 se asocia al foro F_1, con (1, 2, 5, 2) representando a (r_2, e_2, D_2, H_2). El $d_2 = r_2 + D_2$, $d_2 = 6$.
- Cuando se asocia f_2 al foro F_1, el foro ya se encuentra activo. Como cumple con la condición $d_1 < d_2 < DF_1$, entonces realiza el test de viabilidad y como el resultado de $\sum_{i=1}^{m} max(E_{iH1}, E_{iH2}, ..., E_{iHp})/DF_i$ es 2/8; por lo tanto es aceptado en el foro.
- El filósofo f_3 se asocia al foro F_2, con (2, 4, 5, 3) representando a (r_3, e_3, D_3, H_3). El $d_3 = r_3 + D_3$, $d_3 = 7$.
- Cuando se asocia f_3 al foro F_2, activa el foro y se asigna a $DF_2 = d_3 * (1 + GR_2) = 10$. Se realiza el test de viabilidad con respecto a los otros grupos para ver si se lo acepta o rechaza. Al realizar el test se tiene que $E_1 = 2$, $E_2 = 4$, $DF_1 = 8$, $DF_2 = 10$, el valor de $U = 13/20$ por lo tanto es aceptado.
- El filósofo f_4 se asocia al foro F_2, con (3, 2, 4, 4) representando a (r_4, e_4, D_4, H_4). El $d_4 = r_4 + D_4$, $d_4 = 7$.
- Cuando se asocia f_4 al foro F_2, el foro ya se encuentra activo. Como cumple con la condición $d_3 < d_4 < DF_2$, entonces realiza el test de viabilidad y le da como resultado que $U = 13/20$ por lo tanto es aceptado en el foro.

6.- Conclusiones

Con la utilización de algoritmos de planificación de tiempo real no estricto, es posible resolver el problema de exclusión mutua para grupos de procesos con deadlines. En el trabajo se presenta una solución para ambientes centralizados y su extensión a ambientes distribuidos. Inicialmente se considera que la utilización del recurso es concurrente entre todos los procesos que se encuentran en el grupo. Un ejemplo de aplicación, como se presentó en la sección 5.3.1, puede ser el caso de un conjunto de foros que comparten una única sala de conferencia. Cada foro agrupa un conjunto de disertantes, cada uno de ellos necesita un determinado tiempo de exposición en esa sesión, y no todos llegan en el mismo instante de tiempo. El tiempo que utiliza cada foro la sala de conferencia es limitado, para permitir que otros foros puedan ingresar.

En su versión extendida, la utilización del recurso se desarrolla en forma paralela y/o concurrente dentro de cada grupo. Un ejemplo para este caso es el presentado en la sección 5.3.2. y corresponde al conjunto de foros anterior que comparten una misma sala de conferencia, donde los disertantes intervienen todos juntos en un debate. No todos llegan en el mismo instante de tiempo, y el tiempo que utiliza cada foro la sala de conferencia es limitado, para permitir que otros foros puedan ingresar.

El problema de la exclusión mutua está garantizado con la política no apropiativa del algoritmo de planificación de tiempo real, evitando el nivel de competencia de acceso que presenta el modelo del problema original. Se mantiene como política del esquema original el control de permanencia del proceso que activa el grupo, para minimizar el tiempo de respuesta.

Referencias

[1] Wenming Li, Krishna Kavi y Robert Akl. *An Efficient Non-Preemptive Real-Time Scheduling.* Department of Computer Science and Engineering, The University of North Texas, USA.

[2] C. L. Liu, J. W. Layland *Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment.* ACM, 1973.

[3] W.A. Horn. *Some simple scheduling algorithms.* Naval Research Logistics Quarterly, (21), 1974.

[4] Yuh-Jzer Joung *Asynchronous Group Mutual Exclusion (extended abstract).* In Proc. 17 th. ACM PODC, Junio 1998.

[5] Patrick Keane, Mark Moir *A Simple Local-Spin Group Mutual Exclusion Algorithm.*

[6] J. Anderson, Mark Moir *Using local-spin k-exclusion algorithms to improve wait-free object implementations.* Distributed Computing, 11:1-20, 1997. Una versión preliminar apareción en Proceedings of the 13th Annual ACM Symposium on Pronciples of Distributed Computing, 1994, pp. 141-150.

[7] Karina Cenci, Jorge Ardenghi *Exclusión Mutua para Coordinación de Sistemas Distribuidos.* CACIC 2001

[8] Karina Cenci, Jorge Ardenghi *Algoritmo para Coordinar Exclusión Mutua y Concurrencia de Grupos de Procesos* CACIC 2002

[9] K. Vidyasankar *A simple group mutual* l-*exclusion algorithm* Information Processing Letters, 2003.

[10] L. Lamport *A Fast Mutual Exclusion Algorithm* ACM on Transactions on Computer Systems, Volume 5, Number 1, Febrero 1987

[11] Gary L. Peterson. *Myths about the mutual exclusion problem.* Information Processing Letters, Junio 1981.

[12] G. L. Peterson, M. J. Fischer *Economical Solutions for the Critical Section Problem in a Distributed System* ACM Annual proceedings of Theory of Computing, pp.91-97, 1977.